

Dokument nieoficjalny, dla implementatorów.

Wskazówki, wyjaśnienia, głównie do implmentatorów.

Słowem wstępu.

Dokładność:

Ta implementacja ma **imitować** działanie aplikacji, nie przykładajcie się jakoś ultra-bardzo do tego bo to w momencie zaliczenia przedmiotu cała aplikacja leci do kosza, róbcie to z głową i nie tracie za wszelką cenę czasu w święta 🎁, +- to nam powiedział prowadzący. Gdybyśmy mieli pisać to ultra-dokładnie dbać o bezpieczeństwo, brak bugów to pewnie skończyli byśmy w styczniu, ale 2023. Jak coś nie będzie działało, albo będzie jakiś bug to się też jakoś mocno tym nie przejmujcie, róbcie tak, żeby wyglądało, że działa!!!!!! Jedyne co to **do 06.01 mają być widocznie jakieś efekty implementacyjne.**

PS: prowadzący wam w kod nie będzie zaglądał !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Trzymanie się wytycznych architekta:

Staralem się mocno, aby było to zaprojektowane sensownie, ale **nie trzymajcie się tego na sztywno**, uznacie, że coś według was powinno wylecieć bo można jakoś łatwiej coś zaimplementować bądź zaimitować (🤖) to to róbcie i się nie przejmujcie.

Technologie:

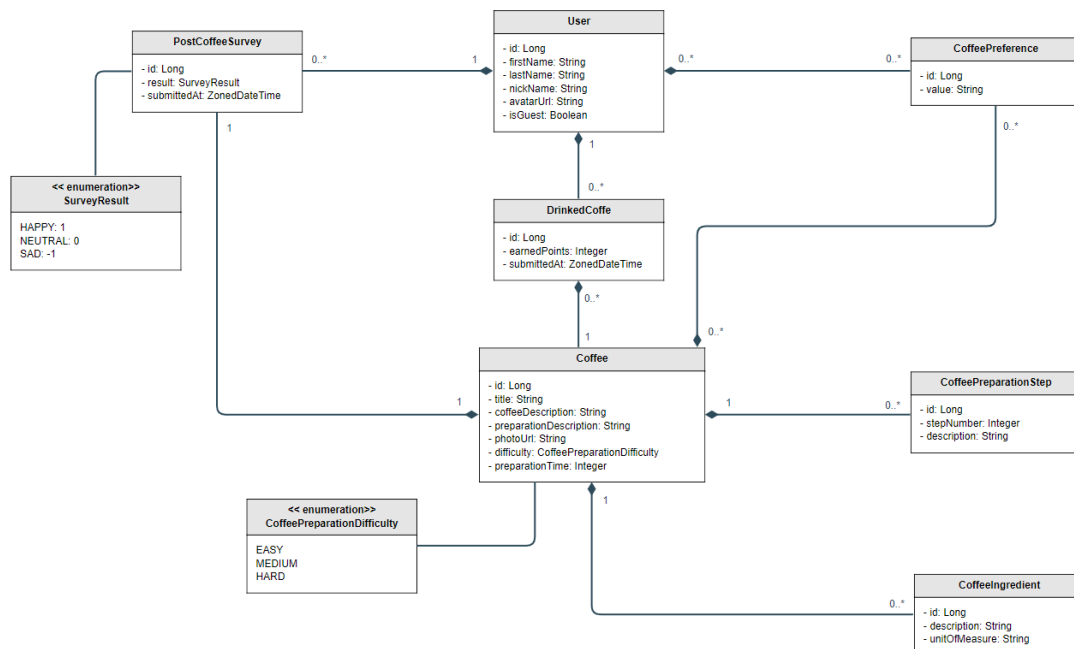
Jeżeli chcecie coś zmienić, to raczej na jestem otwarty na propozycje, ale raczej patrzyłem aby było chociaż to co częściowo macie na Pracowni Programowania (np. Spring) albo co już było na TINach np. Flask.

Hardcodowanie:

Starajcie się takich rzeczy jak hasła do baz danych, czy np URL do webhooka z discorda, nie wpisywać w kodzie, używajcie zmiennych środowiskowych. Pomijam bezpieczeństwo, ale DevOps będzie miał możliwość podłączenia do innej bazy danych, czy wklejenia linka z webhookiem z docelowego serwera na dc.

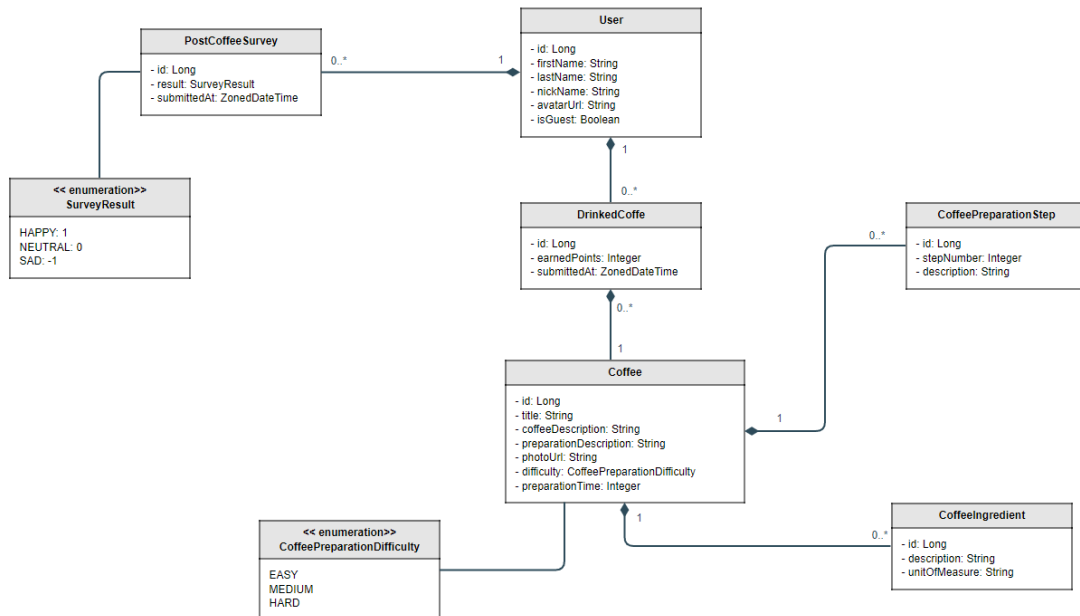
Diagram Klas.

UML Class diagram



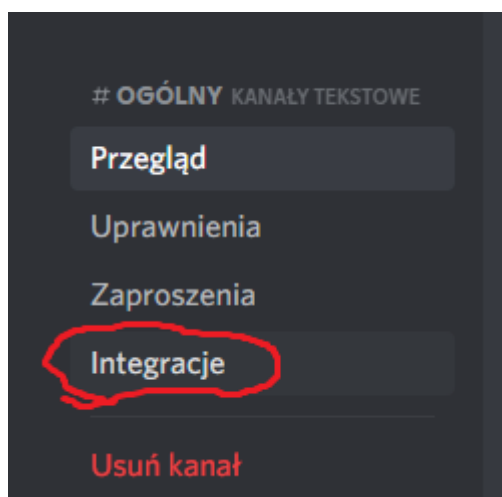
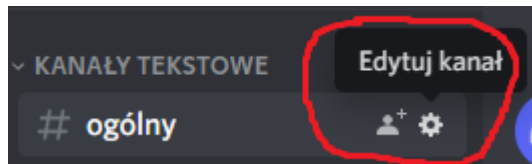
Oficjalnie to powinno wyglądać tak, jednakże jestem świadomy czasu jaki macie na implementację, oraz to, że są święta i sesyjka. Bardzo czasochłonne wydaje się dobieranie kaw na podstawie zgromadzonych danych odnośnie preferencji itp. Stąd sugeruję rozwiązanie tego w taki sposób, aby wprowadzić trochę losowości (🤖) w dobór kaw i wtedy przechowywanie tych preferencji mija się z celem. Dodatkowo, przechowywanie informacji po jakiej kawie poprawił się/pogorszył humor również docelowo może dla nas nie mieć znaczenia. Stąd **nieoficjalny UML dla was** by wyglądał tak:

UML Class diagram



Wysyłanie wiadomości na discordzie.

Polecam stworzyć sobie swój jakiś serwer, albo mieć wymagane uprawnienia, a następnie



OGÓLNY KANAŁY TEKSTOWE

Przegląd

Uprawnienia

Zaproszenia

Integracje

Usuń kanał

Integracje

Spersonalizuj serwer poprzez integracje. Zarządzaj webhookami i śledzonymi kanałami, które publikują na tym kanale. [Dowiedz się więcej na temat zarządzania integracjami.](#)

Webhooki

0 webhooków

Tworzenie webhooka


Śledzone kanały

0 kanałów


Dowiedz się więcej

Nowy webhook

PUBLIKOWANIE NA KANAŁE #OGÓLNY

 nazwa bota

Utworzono 25 gru 2021 przez użytkownika Dawid Korzēpa#9249



NAZWA

nazwa bota

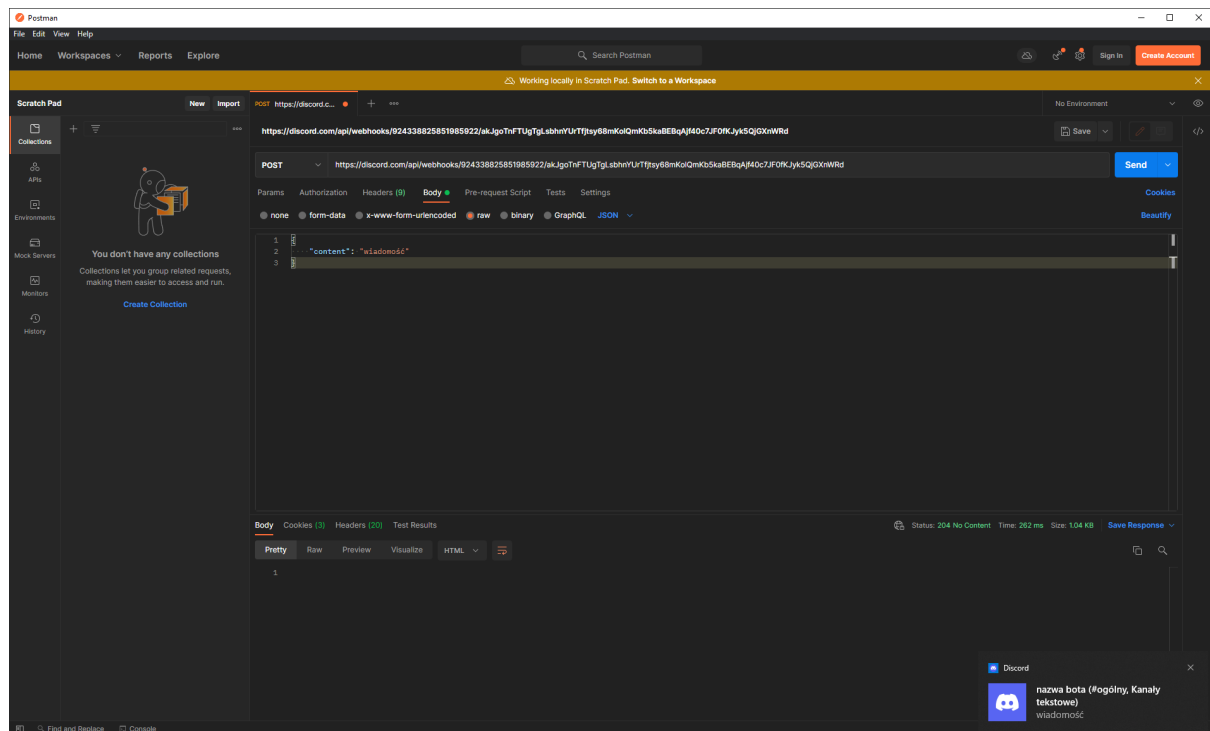
KANAŁ

#ogólny

Minimalny rozmiar: 128x128

Skopiuj adres URL webhooka

Usuwanie webhooka



Wklejamy skopiowany URL i w polu content wpisujemy wiadomość.

Aplikacja we flasku (ta do statystyk).

Jej celem jest prezentacja statystyk z ankiet, oraz umożliwienie filtrowania wiadomości z discorda, dla szefa. Domyślnie aplikacja powinna mieć dwie podstrony, jedną do filtrowania wiadomości, drugą do prezentowania danych z ankiet np. w formie tabelki. Jeżeli komuś się bardzo chce to może zrobić frontend w react/angular/vue, ale sugerowałbym najprostsze rozwiązanie, czyli generowanie szablonów (to samo co na tinach). Do stylowania Bootstrap chyba Bootstrap najprostszy. Dodatkowo, skrypt do zapisywania wiadomości z discorda do Mongo zajmuje całe 20 linijek, z czego 11 to kopiuj-wklej z dokumentacji discord.py. (Służę pomocą tutaj dla deweloperów, więc śmiało się kontaktować).

Hostowanie (sugestie).

Aplikacja na tablety - wygenerujemy .apk i mu wgramy na tablet, z google play za dużo babrania i chyba trzeba zapłacić (?).

REST API w Javie - Heroku, do postawionej aplikacji można dorzucić Postgresa za darmo + automatyczne buildy jak testy przechodzą.

Flask - Heroku, natomiast skrypt do zapisywania wiadomości z dc mogę wam do siebie na AWS wrzucić i będzie 24/7 działało i czuwało nad wiadomościami z dc.

MongoDB - za darmo(dla naszych potrzeb), bez kart można skorzystać z [oficjalnej chmurki mongo](#), baza danych będzie chodziła 24/7, idealne rozwiązanie do zapisywania wiadomości z dc.